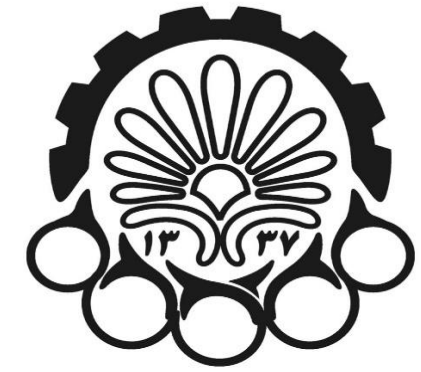

دانشکده مهندسی برق

نام دانشجو: کسری خلفی

شماره ی دانشجویی : ۹۵۲۳۰۳۸

استاد درس : دکتر فرزانه عبداللهی

سرپرست ازمایشگاه: مهندس امینی

آزمایش شماره ی سوم

پایتون:

هدف در این آزمایش تشخیص عملیات XOR به کمک پرسپترون دو لایه میباشد. تعداد لایه ها به صورت دلخواه میباشد که ما در این ازمایش تعداد نورون های ورودی را برابر دو انتخاب کردیم. با توجه به اینکه یک خروجی داریم در نتیجه تعداد نرون خروجی برابر یک میباشد.

تابع فعال ساز لایه ی اول sigmoid و تابع فعال ساز لایه ی دوم تابع خطی میباشد.

کد پایتون چند لایه به صورت زیر میباشد:

```python
import numpy as np
from sigmoid import sigmoid, dSigmoid
import matplotlib.pyplot

error = 0
y = np.array([[1, 1, -1], [1, 0, -1], [0, 1, -1], [0, 0, -1]], dtype=float)
d = np.array([0, 1, 1, 0], dtype=float)
w1 = np.array([[.1, .2, .3], [.2, .3, .1]], dtype=float)
w2 = np.array([[.1, .1]], dtype=float)
a = np.array([0, 0], dtype=float)
counter = 0
step = 0

while True:
    a[0] = sigmoid(np.dot(w1[0], np.transpose(y[counter])))
    a[1] = sigmoid(np.dot(w1[1], np.transpose(y[counter])))
    o = np.dot(w2, a)
    error = error + (o - d[counter]) ** 2
    if counter == 3:
        step += 1
        print("error in {} -> {}".format(step, error))
        if error < .001:
            break
        error = 0
        counter = -1
    F = np.array([[dSigmoid(a[0]), 0], [0, dSigmoid(a[1])]])
    s2 = -0.01 * 1 * (d[counter] - o)
    s1 = np.dot(np.dot(F, np.transpose(w2)), s2)

    w1 = w1 - np.dot(np.transpose(np.array([np.dot(np.dot(F, np.transpose(w2)), s2)])), np.array([y[counter]]))
    w2 = w2 - s2 * a
    counter += 1

```

```python
    counter += 1

print("---------------")
for cnt in range(4):
    a[0] = sigmoid(np.dot(w1[0], np.transpose(y[cnt])))
    a[1] = sigmoid(np.dot(w1[1], np.transpose(y[cnt])))
    o = np.dot(w2, a)
    print("{} -> {}".format(y[cnt], o))

```

با استفاده از این الگوریتم، وزن‌های اولیه و گام آموزش بعد از ۲۵۵۲۶۵ بار feed کردن هر ۴ ورودی به سیستم خطای مجموعمان کمتر از ۰.۰۰۱ خواهد شد و نتایج نهایی بصورت زیر پدید خواهد آمد.

```
[ 1.  1. -1.] -> [0.01594805]
[ 1.  0. -1.] -> [0.9849528]
[ 0.  1. -1.] -> [0.98469863]
[ 0.  0. -1.] -> [0.01573593]
```

دقت شود که -۱ در ورودی‌ها به عنوان bias در نظر گرفته شده و ورودی‌های اصلی دو مقدار اول هر آرایه هست

متلب:

کد متلب چند لایه به صورت زیر میباشد

```
error = 0;
y = [1, 1, -1; 1, 0, -1; 0, 1, -1; 0, 0, -1];
d = [0, 1, 1, 0];
w1 = [.1, .2, .3; .2, .3, .1];
w2 = [.1, .1];
a = [0, 0];
counter = 1;
step = 0;

while true
    a(1) = sigmoid(w1(1,:) * transpose(y(counter,:)));
    a(2) = sigmoid(w1(2,:) * transpose(y(counter,:)));
    o = w2 * a';
    error = error + (o - d(counter)) ^ 2;

    if(counter == 4)
        step = step + 1;
        fprintf('error in %i -> %f\n', step, error);
        if(error < .001)
            break
        end
        error = 0;
        counter = 0;
    end
    counter = counter + 1;

    F = [dSigmoid(a(1)), 0; 0, dSigmoid(a(2))];
    s2 = -0.01 * 1 * (d(counter) - o);
    s1 = F * transpose(w2) * s2;

    w1 = w1 - F * transpose(w2) * s2 * y(counter,:);
    w2 = w2 - s2 * a;
end
```